BetterMart

Database management system Report

Group 138

Samridh Girdhar 2021282

Raunaque Khan 2021278

Problem Statement:

- To design a database and an application for an online retail store system.

Scope of the Project:

- This project aims to build a database management system for a retail application for the customers to buy products online.
- A user would be able to survey through the various products added by the retailer on their website and add them in their cart which can be amended later anytime.
- The user could later checkout with their products in the cart and would be redirected to the payment's portal.
- After the payment the retailer is notified, the order is then shipped by them.
- The real time status of the delivery of order could be tracked by the user, which would be constantly updated by the retailer.

Stakeholders:

- Customers (the people who will buy the products)
- Retailers (the people who are registered on the website as sellers)
- Delivery Partners.
- Admin

Entities:

- **Customer** (USER_ID, FIRST_NAME, LAST_NAME, EMAIL, CONTACT_NUMBER, ADDRESS)
- **Retailer** (RETAILER_ID, FIRST_NAME, LAST_NAME, CONTACT_NUMBER, ADDRESS)
- **Order** (ORDER_ID, PRODUCTS, DELIVERY_STATUS, PAYMENT_STATUS)
- **Delivery Partner** (PARTNER_ID, FIRST_NAME, LAST_NAME, REVIEWS, PREV_ORDERS, CURR_ORDER)

Tech stack:

- **Front-end**: HTML, CSS.
- **Back-end:** Flask, Python, MySQL.

Technical Requirements:

- Ensuring correct accessibility for all the stakeholders in the database.
- Fabricating a well-linked data management system, integrating well between the dbms and website.
- Handling large amount of data.
- Avoiding concurrency issues.
- Ensuring good encryption of data and privacy.
- Delivery guys request handling and reviewing.
- Retailers stocking of new items and categories.
- Customers personalizing their experience of the online retail store.
- Admin managing the Data of all the people involved in the retail, from the delivery partners to the retailers and customers, the admin ensures that every small operation is done satisfactorily.

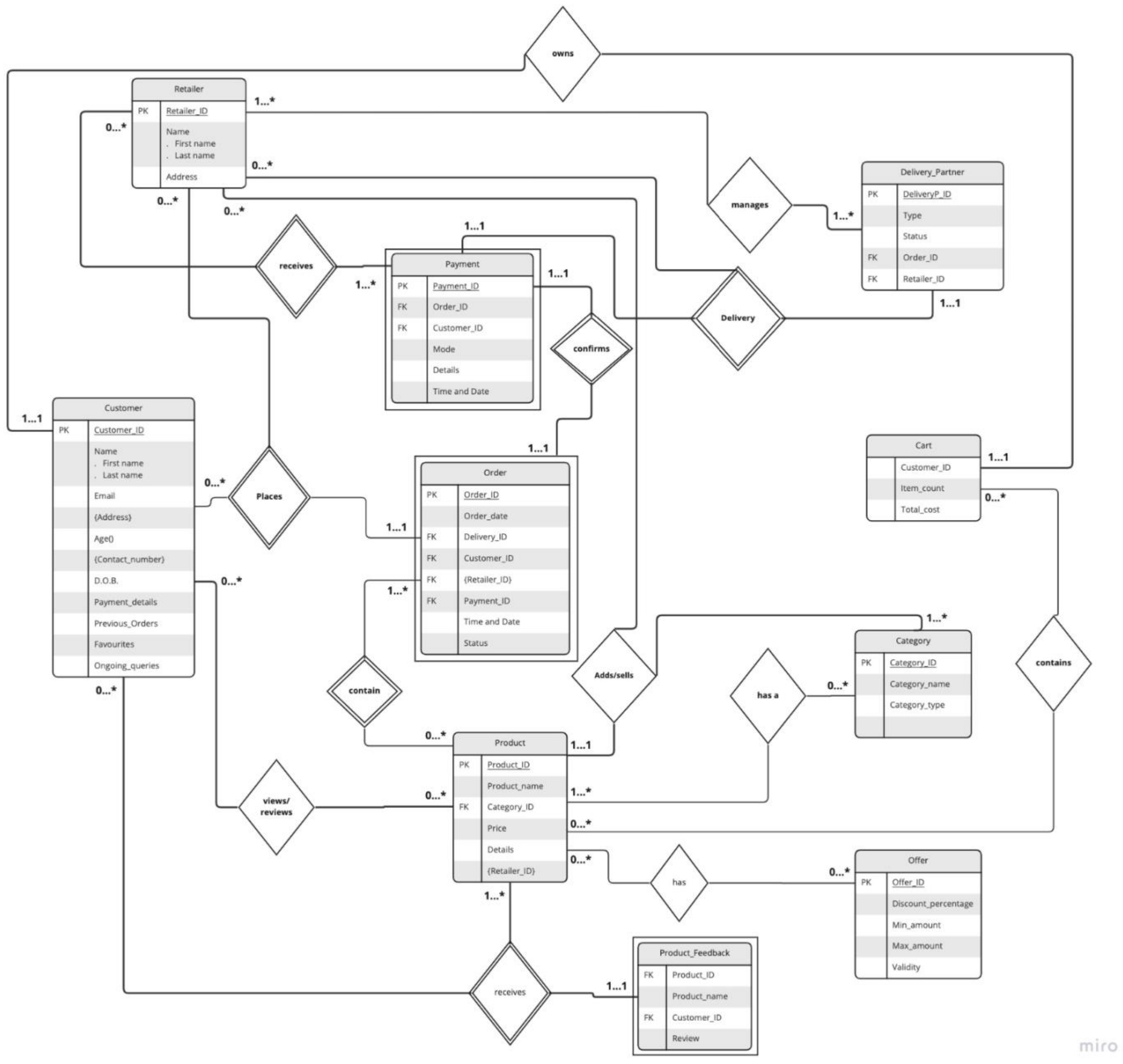
Functional Requirements:

- Customers create their account, and with purchases get access to special offers.
- Adding, removing, replenishing, and updating details of the products that can be added by the retailer.
- Filtering products and sorting based on various parameters can also be done.
- Users can customize their cart however they like.
- Retailers can register themselves on the website and start adding their products, and their stock respectively.
- Different databases will be fabricated in accordance with the required functionalities.
- Payments can be made in whichever way the customer likes. COD is not available beyond a certain amount in the cart.

- Once the payment is done, the transporter and the various other details regarding the tracking of the order is shared with the user.
- Cancelling of the order can also be done, and once the delivery is done, the users would be allowed to drop reviews of the products.
- If the customers would have any questions regarding the functionality of the product, they'd be allowed to drop a message to the retailers.

As Explained in Deadline 2, the database for BetterMart has a total of 10 entities:

1. Customer
2. Product
3. Product Feedback
4. Offer
5. Category
6. Orders
7. Cart
8. Delivery Partner
9. Payment
10. Retailer



- The given database management system has ten tables: cart, category, customer, delivery_partner, offer, orders, payment, product, productfeedback, and retailer.
- Table "cart" has four columns: cart_id, customer_id, item_count, and total_cost. The cart_id is the primary key of this table, and the customer_id is a foreign key that references the customer table's customer_id. This table stores information about the customer's cart, including the number of items and the total cost.
- Table "category" has two columns: category_id and category_name. The category_id is the primary key of this table. This table stores information about the product categories.
- Table "customer" has seven columns: customer_id, customer_name, customer_email, customer_address, customer_age, customer_number, and customer_paymentDetails. The customer_id is the primary key of this table. This table stores information about the customers, including their name, email, address, age, phone number, and payment details.
- Table "delivery_partner" has five columns: delivery_partner_id, type, status, order_id, and retailer_id. The delivery_partner_id is the primary key of this table. The order_id and retailer_id are foreign keys that reference the orders and retailer tables' order_id and retailer_id columns, respectively. This table stores information about the delivery partners, including their type and status.
- Table "offer" has five columns: offer_id, discount_percentage, min_amount, max_amount, and validity. The offer_id is the primary key of this table. This table stores information about the offers, including the discount percentage, minimum and maximum amounts, and validity period.
- Table "orders" has eight columns: order_id, delivery_partner_id, customer_id, retailer_id, payment_id, time_date, status, and cart_id. The order_id is the primary key of this table. The delivery_partner_id, customer_id, retailer_id, payment_id, and cart_id are foreign keys that reference the delivery_partner, customer, retailer, payment, and cart tables' delivery_partner_id, customer_id, retailer_id, payment_id, and cart_id columns, respectively. This table stores information about the orders, including the delivery partner, customer, retailer, payment, time, and status.
- Table "payment" has six columns: payment_id, mode, details, order_id, customer_id, and time_date. The payment_id is the primary key of this table. The order_id and customer_id are foreign keys that reference the orders and customer tables' order_id and customer_id columns, respectively. This table stores information about the payments, including the payment mode, details, and time.
- Table "product" has six columns: product_id, product_name, category_id, product_price, retailer_id, and details. The product_id is the primary key of this table. The category_id and retailer_id are foreign keys that reference the category and retailer tables' category_id and retailer_id columns, respectively. This table stores information about the products, including the name, category, price, retailer, and details.
- Table "productfeedback" has five columns: product_feedback_id, product_id, product_name, customer_id, and review. The product_feedback_id is the primary key of this table, and the product_id is a foreign key that references the product table's product_id. This table stores information about the product feedback, including the product name, customer, and review.
- Table "retailer" has two columns: retailer_id and retailer_name. The retailer_id is the primary key of this table. This table stores information about the retailers, including their name.
- Overall, this database management system stores information about customers, products, orders, payments, and feedback. The tables are linked by primary

Schema Creation:

- In order to create a schema for our online retail store database, We first created a database on MySQL using the command:

  ```
  CREATE DATABASE [database_name]
  ```

- Then we inserted tables into the database for each of the above-stated entities, we did this using the syntax:

  ```
  CREATE TABLE [database_name].[table_name](

          ......
  ```

);

- And then using suitable Datatypes we added the attributes as mentioned in the ER diagram.

Primary Keys:

A primary key is an attribute that acts as a unique Identifier.

We defined the primary keys using:

PRIMARY KEY (col_name)

Primary keys use **AUTO_INCREMENT** and are **NOT NULL**

	Entity Name	Primary Key
1	Customer	Customer_ID
2	Product	product_ID
3	Product Feedback	product_ID
4	Offer	Offer_ID
5	Category	Category_ID
6	Cart	Customer_ID
7	Orders	Order_ID
8	Delivery Partner	DeliveryP_ID
9	Payment	Payment_ID
10	Retailer	Retailer_ID

- Integrity Constraints:

  - Auto_Increment
  - Not Null
  - Primary Key
  - Foreign Key

- Indexes**:**

  Indexes are used to retrieve data from the database very fast. We created Indices using:

  CREATE INDEX idx_name **ON** table_name(col_name);

- DATA POPULATION:

  In order to produce bulk data (as in the case of customers, distributors, offers, etc.), we have used an online bulk data generator.

  The following syntax was used at some points:

  INSERT INTO database.tbl_name(col1,col2) **VALUES** (col1_val, 'col2_val'),

- Database :

  i) In the Back End, we have created a database and would be connected to the front end website through *MySQL*.
  ii) The Database is made such that it could easily handle large amounts of data and give concurrent results without performance degradation.
  iii) Throughout the database, atomicity has been assured as to reduce conflicts due to large amounts of entries.